WWW.SCIAM-MAGAZINE.NAROD.RU

Программы, генерирующие узоры для обоев

А. К. ДЬЮДНИ

ОБЫЧНЫЕ обои печатаются вращающимся цилиндром, на котором выгравирован узор. При каждом обороте цилиндра отпечатывается одно и то же изображение. А вот компьютер может воспроизводить затейливые узоры, которые я называю «обоями для ума» и которые, вообще говоря, не повторяются или во всяком случае повторяются не полностью. Изображение все время проявляется в новых конфигурациях, слева и справа, внизу и вверху. Интересно понаблюдать за тем, что изменяется и что сохраняется от одной картинки к другой.

Цветные образчики в моей коллекции демонстрируют результаты, полученные с использованием трех существенно различающихся методов. Компьютерные программы, при помощи которых были получены изображения, не одинаковы по сложности — среди них есть и простые, и не очень сложные. Они принадлежат трем нашим читателям: Дж. Коннету из Миннесотского университета, Б. Мартину из Бирмингема (Англия) и Т. Смиту из Эсседона (Австралия).

Программа Коннета основана на окружности, но в ее причудливых узорах проступают и квадраты. Ввиду ее двойственной природы мне пришлось назвать ее CIRCLE^2 (КРУГ^2). В нескольких словах, в ней применяется формула окружности $x^2 + y^2$, для того чтобы придать тот или иной цвет точке с координатами x и y. Подробности я опишу ниже. А пока замечу, возможно, к вашему удивлению (а поначалу и к моему), что в обоях подобного рода таится нечто большее, чем множество концентрических окружностей: по мере того как вы отходите назад от стены, все более отчетливо могут проступать на ней изящные структуры квадратной формы (см. рисунки на с. 93).

Наверное, не стоит удивляться тому, что программа CIRCLE^2 родилась под влиянием множества Мандельброта, о чем речь шла в октябрьском номере нашего журнала за 1985 г. Открытие этого множества принадлежит Б. Мандельброту из исследовательского центра Томаса Уотсона фирмы IBM. Богатство формы и красок в графическом изображении множеств Мандельброта основано на единственной математической функции, применяющейся повторно к собственному результату для каждого комплексного числа из заданной области на плоскости. Всякий раз, когда итерируемое значение функции достигает абсолютного значения 2, количество итераций, необходимых для достижения этого значения, определяет цвет соответствующей точки.

Коннет не располагает цветным монитором, поэтому он решил поставить черный цвет в соответствие точкам, для которых абсолютная величина достигает 2 после четного числа итераций, и белый — точкам, для которых количество итераций нечетно. Таким образом, Коннету удалось получить достаточно хорошие изображения множества Мандельброта, но он этим не удовлетворился и занялся исследованием других формул. В конце концов он остановился на формуле $x^2 + y^2$ и отказался от итераций. Его программа систематически сканирует решетчатую область на плоскости. Для каждой точки (x, y) она вычисляет значение формулы и выделяет целую часть из полученного результата. Если это целое число оказывается четным, точка (x, y) окрашивается в черный цвет, если нечетным, то в белый (остается неокрашенной).

Боюсь, что я уже потерял половину своей аудитории. Читатели, наверное, поняли, как устроена программа, и бросились к своим компьютерам, чтобы ввести ее в память машины. Конечно же, программа очень проста. В алгоритмической записи программа CIRCLE^2 состоит из секции ввода, за которой следует двойной цикл:

```
input corna, cornb
input side
for i ← 1 to 100
for j ← 1 to 100
    x ← corna + (side × i/100)
    y ← cornb + (side × j/100)
    z ← x^2 + y^2
    c ← int(z)
    if c is even, then plot (i, j)
```

Сначала программа ждет ввода координат (*corna, cornb*) левого нижнего угла исследуемого квадрата. Переменная *side* представляет длину стороны этого квадрата. Например, если ввести значения -15 и -20 в качестве координат левого нижнего угла квадрата и 87 как длину его стороны, то программа изобразит массив точек 100×100, заключенный в квадратной области со стороной 87 единиц, левый нижний угол которой будет находиться в точке с координатами $(-15, -20)$. В моей версии программы индексы циклов пробегают значения от 1 до 100, однако эти пределы следует выбирать таким образом, чтобы квадрат лежал в пределах экрана того монитора, которым вы располагаете. На моем мониторе при выбранных предельных значениях квадрат получается довольно маленьким.

Двойной цикл проходит через квадратную решетку и для каждой пары индексов (i, j) вычисляет координаты точки (x, y), которой соответствует эта пара. Затем в теле цикла x и y возводятся в квадрат, сумма полученных квадратов присваивается переменной z и из этого значения извлекается целая часть. Наибольшее целое число, большее или равное сумме квадратов x и y, хранится как значение переменной c. Если c делится на 2, то точка (x, y) высвечивается на экране в виде цветного элемента изображения или как темная точка при выводе картинки на печатающее устройство. Если значение переменной c оказывается нечетным, то соответствующая точка не высвечивается или не печатается.

Читателям, которые пожелают воспроизвести обои Коннета, не следует думать о том, правильно ли они поняли, каким цветом нужно изображать «четные» и «нечетные» точки. Большинство рисунков не утрачивают свою красоту при выборе любых цветов. На самом деле количество цветов может быть больше двух. Вместо того чтобы определять, четно или нечетно значение переменной c, его можно делить на число, соответствующее количеству цветов, которым вы располагаете. Тогда величина остатка от деления будет определять тот или иной цвет. Например, чтобы получить изображения, приведенные на с. 93, были выбраны соответственно два цвета для верхнего узора, три для среднего и четыре для нижнего.

Чем меньше исследуемый квадрат, тем плоскость кажется наблюдателю ближе и тем сильнее «увеличение», под которым рассматривает изображение программа CIRCLE^2. Однако в отличие от процедуры раскрашивания окрестности множества Мандельброта программа Коннета не содержит в себе бесконечной регрессии все уменьшающихся узоров. При боль-

WWW.SCIAM-MAGAZINE.NAROD.RU

шом увеличении в окрестности начала координат (0, 0) появляется семейство концентрических окружностей. При еще большем увеличении в середине экрана возникает черный диск; целая часть суммы квадратов координат для каждой точки диска равна нулю. Затем уже весь экран становится темным.

Обои Коннета выглядят значительно интереснее при уменьшении увеличения, когда вы как бы отходите от стены. Концентрические круги будто растворяются в замысловатом узоре первичных и вторичных окружностей, напоминающих муаровые кружева. По мере снижения степени увеличения, как по мановению волшебной палочки, появляются все новые и новые отличающиеся от предыдущих узоры. Может быть, здесь все же скрывается бесконечная регрессия? Я уверен, что читатели сумеют понять, в чем тут дело, а через три месяца я опубликую наиболее интересные ответы.

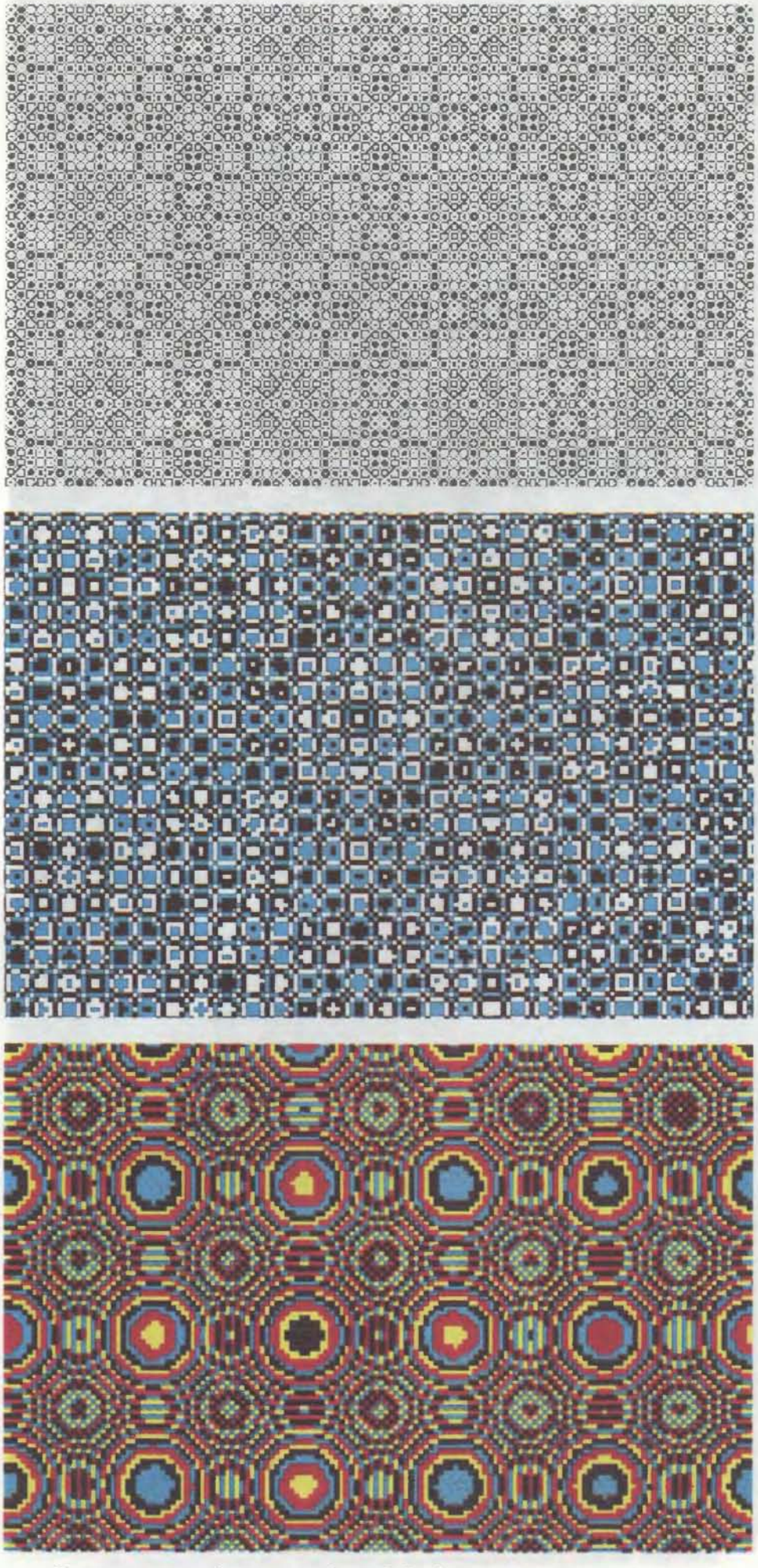
Б. Мартин из Астонского университета в Бирмингеме был также вдохновлен свойствами множества Мандельброта. Он принял на вооружение идею Мандельброта итерировать формулу от начального числового значения — зародыша, но на этом сходство их методов заканчивается. В то время как узоры Мандельброта возникают из комплексных чисел, обои Мартина основаны на итерациях, применяемых к обычным действительным числам. Более того, числовые зародыши для множества Мандельброта представляют собой бесчисленные точки, принадлежащие заданной области на плоскости, а программа Мартина выращивает свои узоры из одного-единственного зародыша.

Мартин предлагает, например, следующие две формулы, которые могут породить потрясающие, отличающиеся довольно сложным узором изображения (см. фото на обложке журнала, а также рисунки на с. 94, 95 и 96):

$$x \leftarrow y - \text{sign}(x) \times [\text{abs}(b \times x - c)]^{1/2}$$
$$y \leftarrow a - x$$

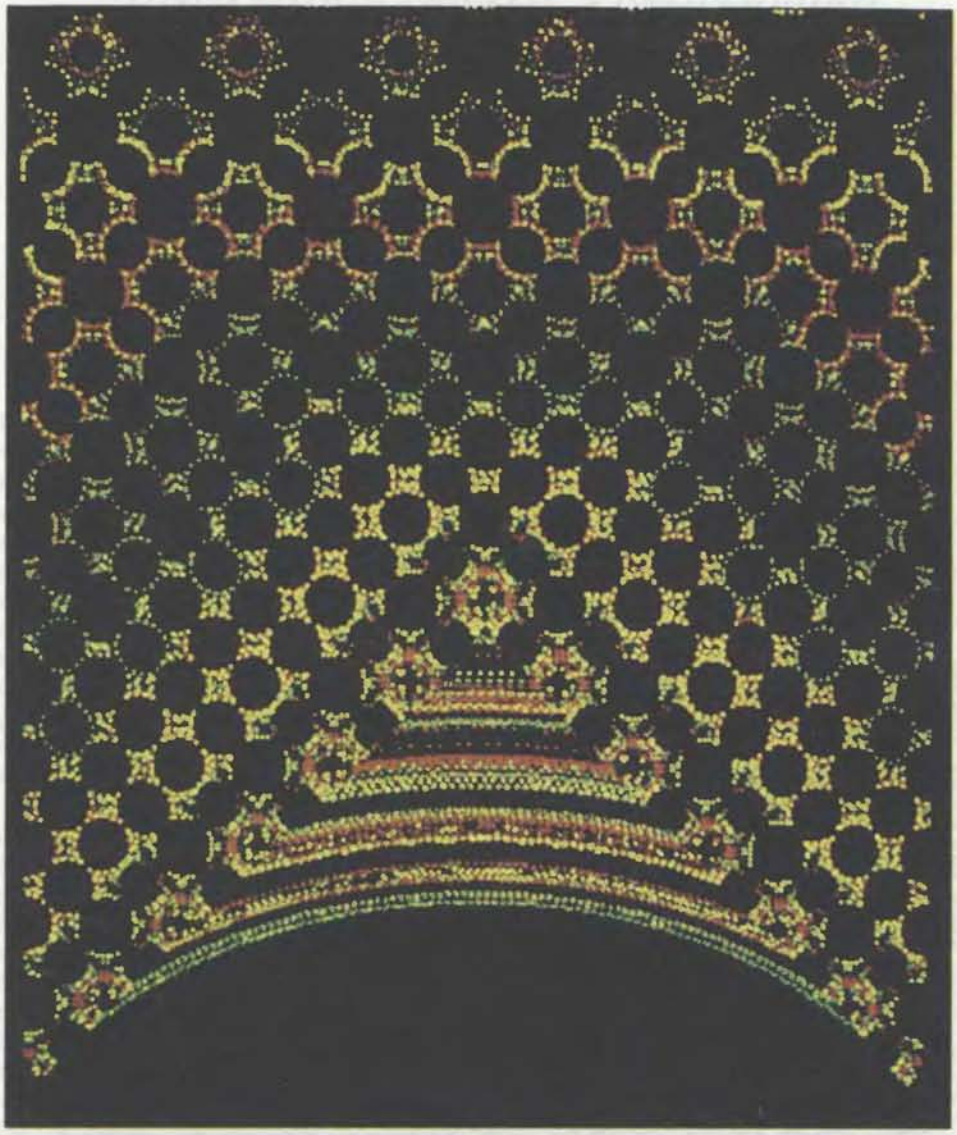
Здесь функция sign(x) принимает значения 1 или -1 в зависимости от того, положителен или отрицателен аргумент x. Функция abs($b \times x - c$) дает абсолютную величину выражения $b \times x - c$. Изображения могут варьировать довольно значительно в зависимости от выбираемых значений a, b и c — числовых параметров в формуле.

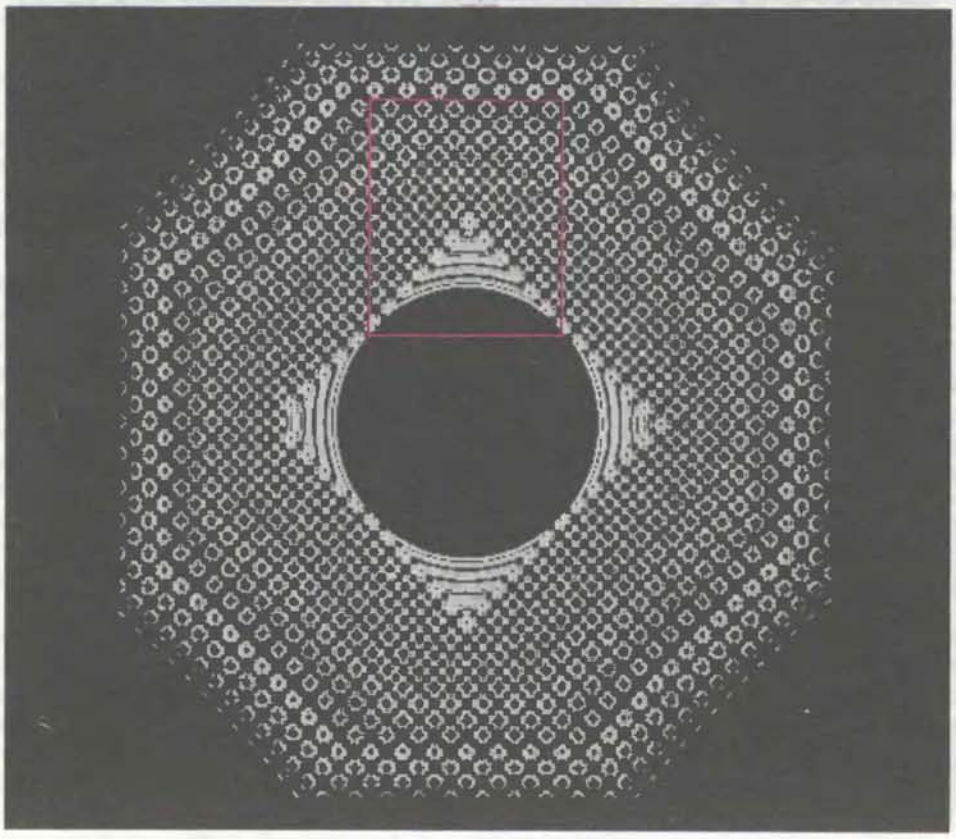
Сами формулы записаны в сокращенной математической форме. Здесь предполагается, что каждый раз новые значения x и y вычисляются в пра-



Окружности и квадраты по модулям 2, 3 и 4, полученные по программе Дж. Коннета CIRCLE[2]

WWW.SCIAM-MAGAZINE.NAROD.RU





«Фрактальные» обои, порожденные программой Б. Мартина HOPALONG

вой части формул и присваиваются x и y в левой части, после чего эти значения снова подставляются в правую часть для вычисления следующих x и y. Таким образом, программа, которую я назвал HOPALONG (попрыгунья), скачет от одной точки к другой. Процесс начинается в точке, где и x, и y равны 0, т. е. в начале координат. Следующая точка может уже располагаться справа вверху, а точка за ней — слева внизу. Компьютер высвечивает точки настолько быстро, что у наблюдателя возникает ощущение миниатюрного электронного дождя, падающего на экран; на нем появляются сотни, а затем и тысячи точек. Вскоре начинает вырисовываться узор. Если, например, параметр a принять равным -200, b равным 0,1 и c равным -80, то возникает восьмиугольный узор (см. нижний рисунок слева). Если рассматривать изображение под увеличением и раскрашивать точку в зависимости от количества скачков, требующихся, чтобы достичь ее, то изображение превращается в удивительные вензеля (см. верхний рисунок слева). Для других значений a, b и c можно получить новые узоры. Можно принять, например, $a = 0{,}4$, $b = 1$ и $c = 0$ (см. верхний рисунок на с. 95) или принять $a = -3{,}14$, $b = 0{,}3$ и $c = 0{,}3$ (см. нижний рисунок на с. 95).

Алгоритм программы HOPALONG почти так же прост, как и программы CIRCLE[2]:

input *num*
input *a, b, c*

$x \leftarrow 0$
$y \leftarrow 0$
for $i \leftarrow 1$ to *num*
plot (x, y)
$xx \leftarrow y - \text{sign}(x) \times [\text{abs}(b \times x - c)]^{1/2}$
$yy \leftarrow a - x$
$x \leftarrow xx$
$y \leftarrow yy$

И опять множество читателей бросились, наверное, к своим компьютерам, чтобы тотчас ввести и запустить программу. Те же, кто останутся с нами, будут вознаграждены более подробным описанием программы Мартина и познакомятся также с третьей программой, производящей рисунки для обоев.

Перед выполнением программы HOPALONG нужно ввести количество итераций, которое присваивается в качестве значения переменной *num*. Затем нужно ввести также значения параметров a, b и c. Чем больше величина *num*, тем более тонким и замысловатым становится узор. Если, например, *num* принять равной

WWW.SCIAM-MAGAZINE.NAROD.RU

10 000, программа высветит на экране 10 000 точек, но для некоторых значений a, b и c это лишь только начало. При $a = -1000$, $b = 0{,}1$ и $c = -10$ и небольших увеличениях изображение напоминает корку четырехдольного лимона (см. нижний рисунок на с. 96). По мере того как число итераций возрастает от 10 000 точек до 100 000 и затем до 600 000, кружева становятся все более и более тонкими и красивыми (см. верх рисунка на с. 96).

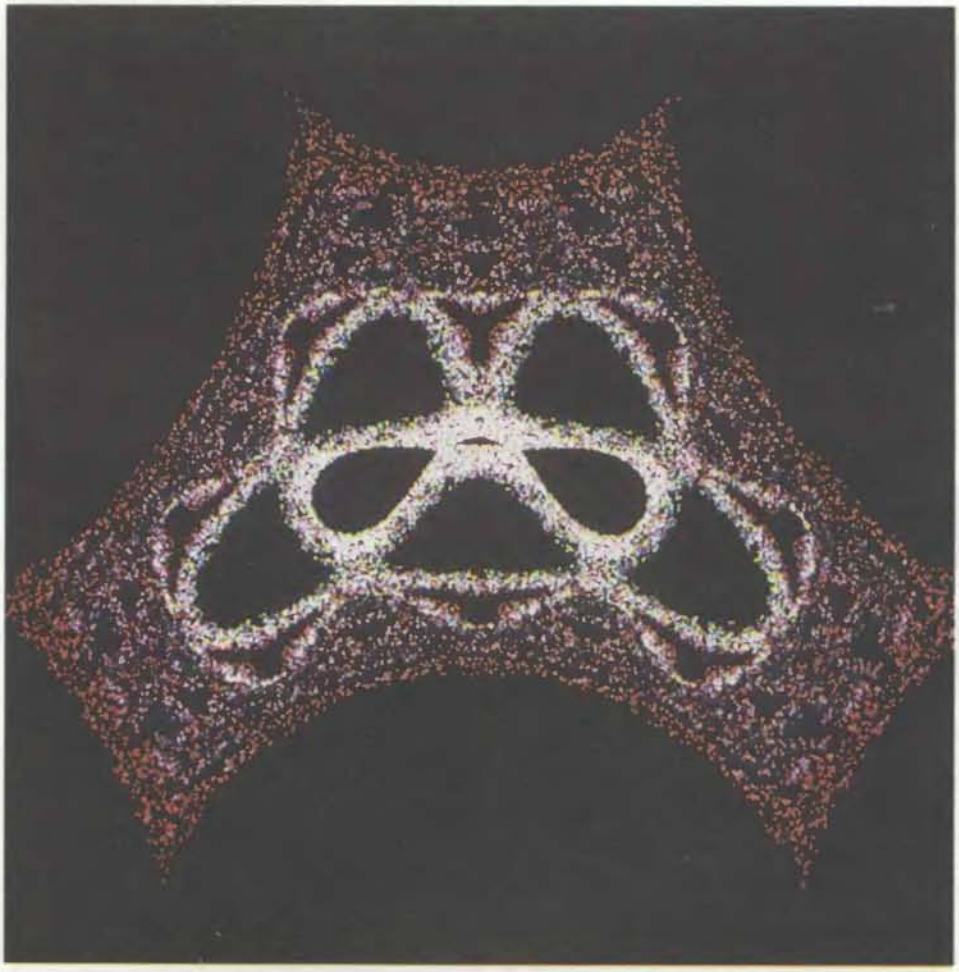
Этот алгоритм можно несколько усовершенствовать. Например, в него можно ввести средство, обеспечивающее перемещение точек, не попадающих на экран, или сжатие внеэкранных областей, чтобы сделать их видимыми. Если в алгоритм введены такие средства, то перед началом выполнения программы следует задать еще три дополнительных параметра, определяющих положение и масштаб картинки. В этом случае тело главного цикла нужно модифицировать: сразу после вычисления x и y в усовершенствованной версии программы HOPALONG осуществляются сдвиг полученных значений x и y и умножение результатов на масштабный множитель.

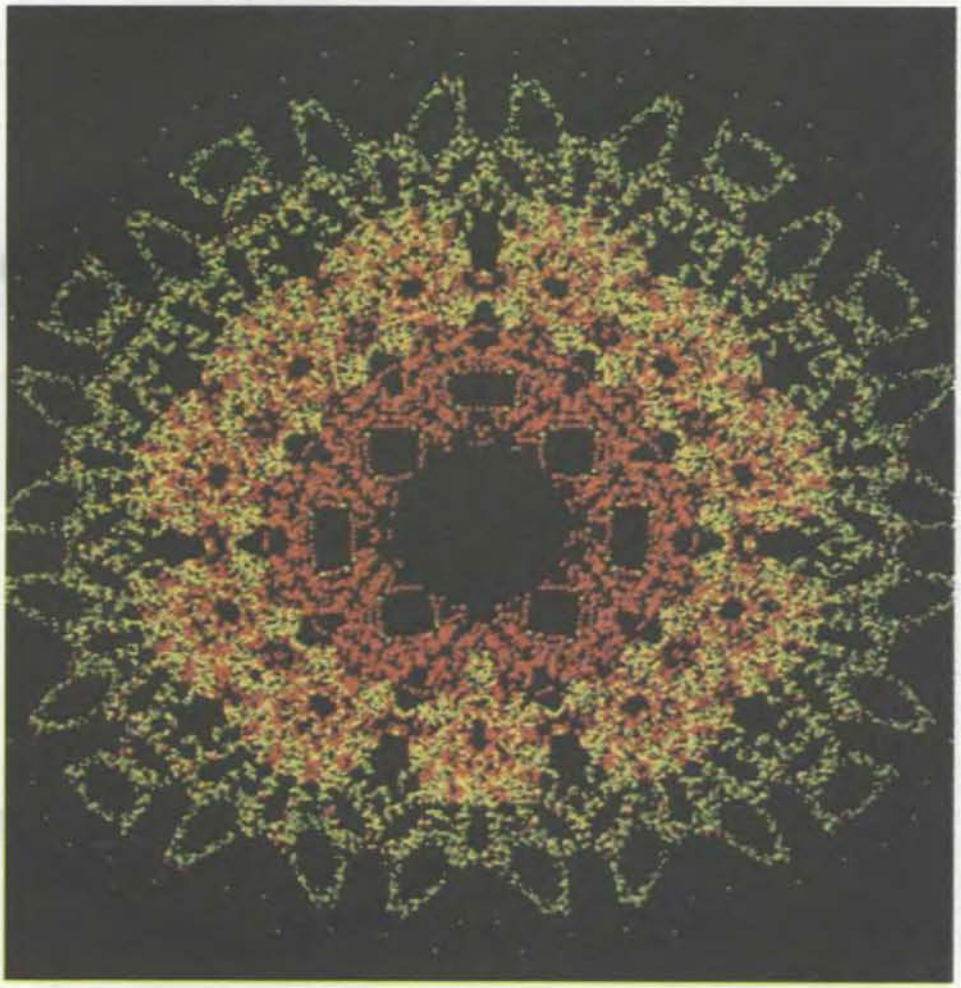
Аналогия с обоями не укрылась и от Мартина: «Мне кажется, что мы находимся на пороге взрыва в области создания узоров, имеющей важное коммерческое значение. Возможно, мы станем свидетелями создания автоматизированного дизайнера, художника по узорам на обоях и тканях, — это может произойти уже в ближайшие несколько лет. Клиент будет сам подбирать себе рисунок по вкусу, задавая компьютеру несколько числовых параметров». Мартин оптимистически смотрит и на возможность применения подобных программ в математической биологии. Посмотрим-ка снова на четырехдольный лимон. При сильном увеличении мы можем видеть детали, весьма напоминающие сосудистую систему; а может быть, это поперечное сечение внешнего слоя односемядольного растения? Об этих и о других узорах Мартин пишет следующее: «Очевидно, эти любопытные конфигурации показывают нам, что правила, по которым строятся сложнейшие ткани живых организмов, могут быть до абсурда простыми».

Возможно, читатели заинтересуются исследованием образов, порождаемых двумя другими итерационными формулами, также предложенными Мартином:

$x \leftarrow y - \sin(x)$ и
$y \leftarrow a - x$

В этих формулах требуется указать





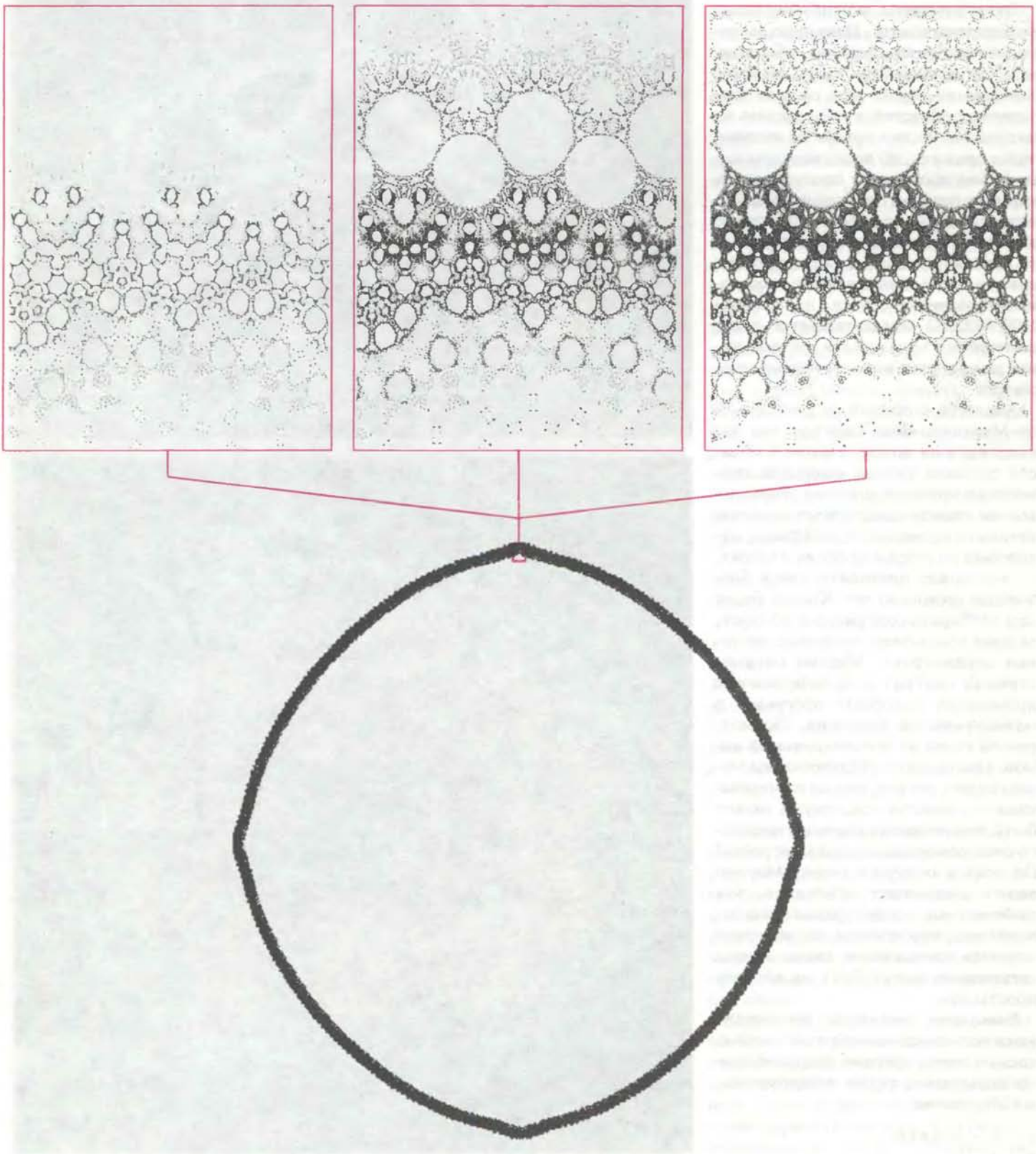
Магические символы, полученные по итерационной формуле Мартина при параметре, равном 7

WWW.SCIAM-MAGAZINE.NAROD.RU

значение лишь одного параметра *a*. Мартину удалось обнаружить ряд в высшей степени интересных узоров, когда *a* лежит в окрестности 0,07 от числа π.

Обои третьей разновидности следует приберечь для комнат, отведенных для глубоких и серьезных размышлений. Узоры здесь самые разные, от персидских хитросплетений до неистовых образов южноамериканских индейских культур (см. нижний рисунок на с. 97 и на с. 98). Методы, при помощи которых были получены эти образы, абсолютно непохожи на рассмотренные выше. Т. Смит из австралийской фирмы PICA Pty. Ltd. разработал тонкие вариации самовоспроизводящегося клеточного автомата, изобретенного в 1960 г. Э. Фредкиным из Массачусетского технологического института (см. статью Б. Хейеса в рубрике «Занимательный компьютер» в декабрьском номере журнала за 1983 г.). В идее этого автомата кроется удивительная возможность для порождения разнообразных узоров, и Смит начал исследования этих огромных возможностей.

Что собой представляет клеточный автомат Фредкина? Представьте себе бесконечную двумерную решетку из



Модель стебля растения и его сосудистая система, порожденные программой HOPALONG

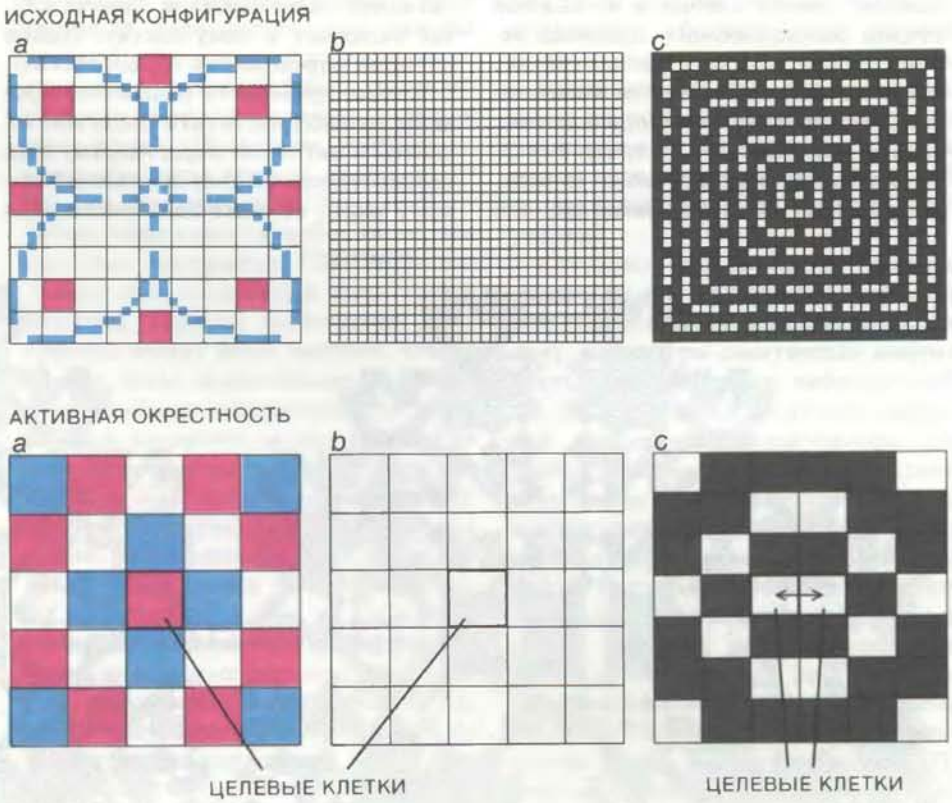
WWW.SCIAM-MAGAZINE.NAROD.RU

квадратных клеток. В каждый данный момент времени каждая клетка может находиться в одном из двух возможных состояний: она может быть либо живой, либо мертвой. Система управляется воображаемыми часами, при тактовых сигналах которых могут меняться состояния клеток. Судьба каждой клетки определяется окружающими ее четырьмя соседними клетками: если число живых соседей на одном такте часов четно, то на следующем такте данная клетка будет мертвой независимо от ее предыдущего состояния. Если же число живых соседей оказывается нечетным, то на следующем такте клетка будет живой. Это правило применяется одновременно ко всем клеткам решетки.

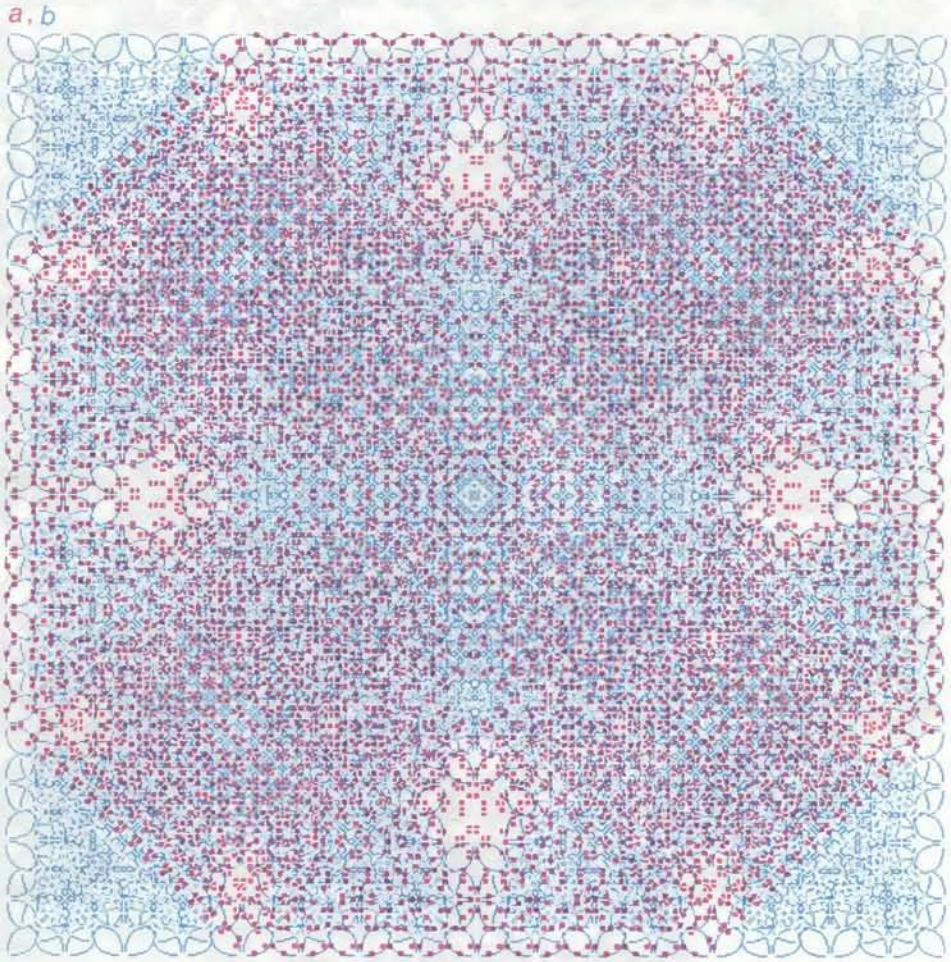
Автомат Фредкина очень напоминает игру «Жизнь», изобретенную Дж. Конвеем из Кембриджского университета. Эту игру мы не раз обсуждали в статьях нашей рубрики. Однако автомат Фредкина появился на свет раньше игры Конвея и он намного проще. Более того, он обладает одним удивительным свойством, отсутствующим у игры «Жизнь», — любая исходная конфигурация живых клеток, проходя через ряд поколений (тактов часового механизма), производит четыре своих копии. Еще через несколько поколений таких копий насчитывается уже 16, затем 64 и т. д. Обои самых интересных образцов появляются в течение жизни промежуточных поколений клеток на стадии создания новых копий исходных популяций живых клеток.

Программа Смита, печатающая обобщенные обои Фредкина, обладает весьма универсальными свойствами и называется PATTERN BREEDER (генератор узоров). Правила, определяющие судьбу клетки, не обязательно зависят от состояния четырех соседних клеток. Перед выполнением программы можно задать конфигурацию окружающих клеток, или «активную окрестность» для каждой клетки. Затем уже программа применяет то же самое правило «четный-нечетный», которое было выбрано в исходной версии автомата Фредкина. Каждый раз, когда на очередном такте часов число живых клеток в активной окрестности данной клетки четное, эта клетка должна умереть в следующем поколении. В противном случае она будет живой.

PATTERN BREEDER работает с любой исходной конфигурацией клеток, заданной пользователем. Например, исходная конфигурация, обозначенная буквой *a* на верхнем рисунке справа, создает красную часть узора, показанного на рисунке внизу. Для



Правила, порождающие клеточные автоматы в программе Т. Смита PATTERN BREEDER



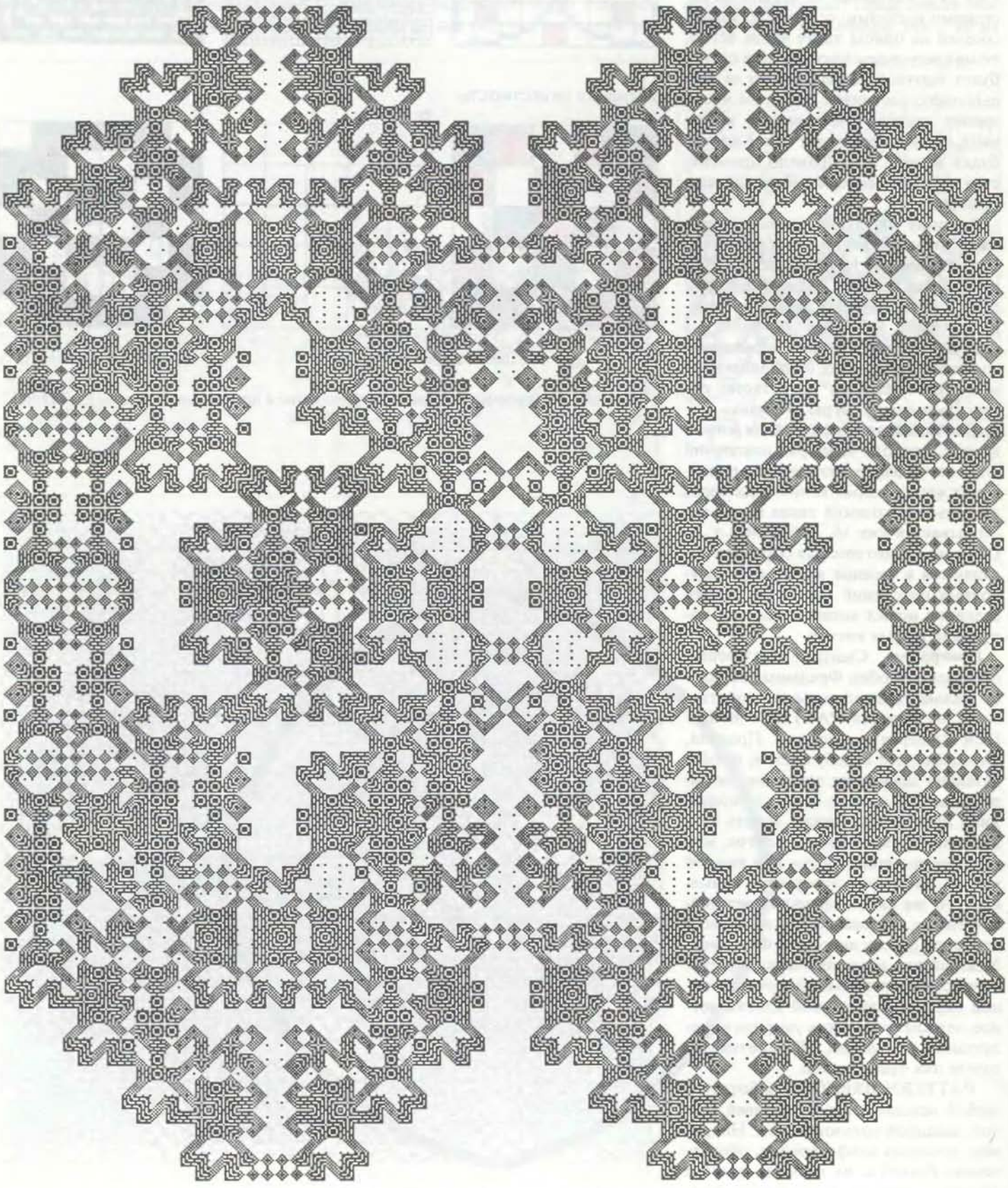
Два наложенных друг на друга изображения, порожденные программой PATTERN BREEDER

WWW.SCIAM-MAGAZINE.NAROD.RU

каждой данной клетки и на каждой стадии эволюционного процесса активная окрестность остается той же. Сама по себе она является довольно сложным узором, в который входят все цветные клетки в матрице 5 × 5, также обозначенной буквой *a* на верхнем рисунке на с. 97. Заметим, что активная окрестность в данном случае включает и саму клетку, судьба которой определяется окрестностью.

Чтобы применить решающее правило, нужно подсчитать число живых клеток в активной окрестности; если рассматриваемая клетка в данный момент жива, ее тоже надо учесть при подсчете. Исходные конфигурации и соответствующие каждой из них активные окрестности показаны также для двух других изображений. Пара, обозначенная буквой *b*, соответствует синей части изображения, приведенного на нижнем рисунке на с. 97, а пара, обозначенная буквой *c*, соот-

c



Абстракция в стиле майя, полученная программой Смита

WWW.SCIAM-MAGAZINE.NAROD.RU

ветствует рисунку на с. 98 . Активная окрестность *c* обладает дополнительным усложняющим свойством: клетка, судьба которой определяется окрестностью, сама осциллирует в последующих поколениях слева направо и обратно в центр окрестности. Программа Смита содержит в себе библиотеку исходных конфигураций и активных окрестностей, из которых пользователь может выбирать интересующие его варианты. В настоящее время программа может выполняться только на персональных компьютерах Apple фирмы Macintosh.

Я не буду описывать программу PATTERN BREEDER во всей ее сложности, но приведу здесь более простую программу под названием FREDKIN. Читатели, обладающие некоторой подготовкой в программировании, смогут затем превратить эту программу в более общий алгоритм, обладающий кое-какими свойствами PATTERN BREEDER Смита:

```
  Ввести исходную конфигурацию
S для каждой клетки массива на экране
  Count ← 0
для каждого соседа этой клетки
  если сосед живой
  то count ← count + 1
  если значение count четно
  то cell ← 0
  иначе cell ← 1
            Вывести cell
Ввести go
переход на S
```

Одна из привлекательных особенностей алгоритмической записи заключается в том, что в вашем распоряжении имеется много различных уровней описания. Конечно, грань, отделяющая общность описания от безответственной туманности, очень тонка. Читатели, наверное, заметят, что описание программы FREDKIN в том виде, в каком оно было приведено выше, отличается большей общностью по сравнению с алгоритмами, рассмотренными ранее. Например, для выполнения инструкции «Ввести исходную конфигурацию» потребуется указание нескольких инструкций на любом из распространенных языков программирования. Эти инструкции будут объединены в двойной цикл с двумя индексами *i* и *j*. Еще один двойной цикл скрывается за инструкцией «Для каждой клетки массива на экране». Здесь два индекса соответствуют координатам точек на экране дисплея или на принтере.

В своем главном цикле программа FREDKIN просто реализует применение правила, управляющего эволюцией заданной конфигурации. Она подсчитывает количество живых членов активной окрестности для каждой клетки (i, j) и затем, если это количество оказывается нечетным, высвечивает или выводит на печать клетку в виде точки. Последняя инструкция главного цикла требует ввода значения для переменной *go*. Здесь пользователь может ввести любое число, после чего программа FREDKIN построит свой следующий узор. Таким образом, процесс выполнения программы может быть прерван, чтобы можно было внимательно рассмотреть особенно понравившийся узор. Данный алгоритм, может быть, и не отвечает полностью требованиям структурного программирования, но работает прекрасно. Иногда, как мы видим, инструкции *GOTO* и связанные с ними уловки оказываются все же полезными.

Через три месяца я опубликую наиболее интересные образцы обоев для ума, присланные к тому времени нашими читателями. Остальные же войдут в мою коллекцию, которую в виде книги я представлю вам, если когда-нибудь вернусь к этой теме.

В СТАТЬЕ, посвященной компьютерной магии, опубликованной в предыдущем номере журнала, внутренняя логика двух трюков осталась необъясненной. Первый из них был основан на известном фокусе Х. Лорейни, нью-йоркского мага. Доброволец из публики вытаскивает карту из хорошо перемешанной колоды и возвращает ее на место. Затем он сдает карты в шесть стопок и находит ту стопку, которая содержит вытянутую им ранее карту. Компьютер повелевает добровольцу сдавать по одной карте из этой стопки. После того как сдана очередная карта, компьютер указывает, следует ли продолжать сдавать или остановиться, угадывая тем самым выбранную карту в сдаваемой стопке.

Каким же образом компьютер останавливается на нужной карте? Ответ заключается в том, что, когда человек сдает колоду на шесть стопок, из центральной части исходной колоды в каждую стопку попадает по одной карте из каждых шести. Каждая из шести карт, находившихся в середине исходной колоды, занимает одно и то же относительное положение в маленьких стопках, а именно она всегда будет четвертой. Поскольку человек сдает карты только из стопки, содержащей выбранную карту, компьютер всегда оказывается в состоянии угадать ее, остановившись после того, как сдана четвертая карта.

Второй нераскрытый в прошлый раз трюк основан на выборе цифр на клавиатуре обыкновенного карманного калькулятора. Доброволец из публики вводит два трехразрядных числа, выполняет над ними операцию умножения и называет все разряды результата, кроме одного. Каким образом факиру удается отгадать этот разряд?

Согласно инструкциям факира, каждое трехразрядное число, выбранное добровольцем, должно представлять собой три десятичных цифры, взятых либо со строки, либо со столбца, либо с главной диагонали цифровой клавиатуры калькулятора. Но любое такое число обладает мистическим свойством «трóечности» после выбрасывания девяток. Когда все разряды числа складываются между собой, их сумма может представлять одно- или двухразрядное число. Если сумма является двухразрядной, то сумма самих разрядов дает одноразрядное число. В любом случае окончательный одноразрядный результат кратен трем. Например, число 471, набранное на клавишах первой колонки клавиатуры, сводится к 12, а затем и к 3, которое, разумеется, кратно 3. В то же время, когда человек перемножает два трехразрядных числа, обладающих описанными свойствами, и произведение подвергается той же самой процедуре, результат получается равным 9. Эта процедура равносильна перемножению по модулю 9 — отсюда мой термин «выбрасывание девяток». Если отсутствует какой-нибудь разряд произведения, скажем x, то в результате отбрасывания 9 для остающихся разрядов имеем $9 - x$. Это число вычисляется магом по мере того, как человек зачитывает все, кроме одного, разряды произведения. Таким образом, по известной величине $9 - x$ можно немедленно указать значение x.